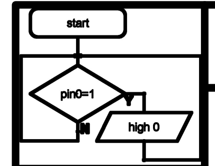
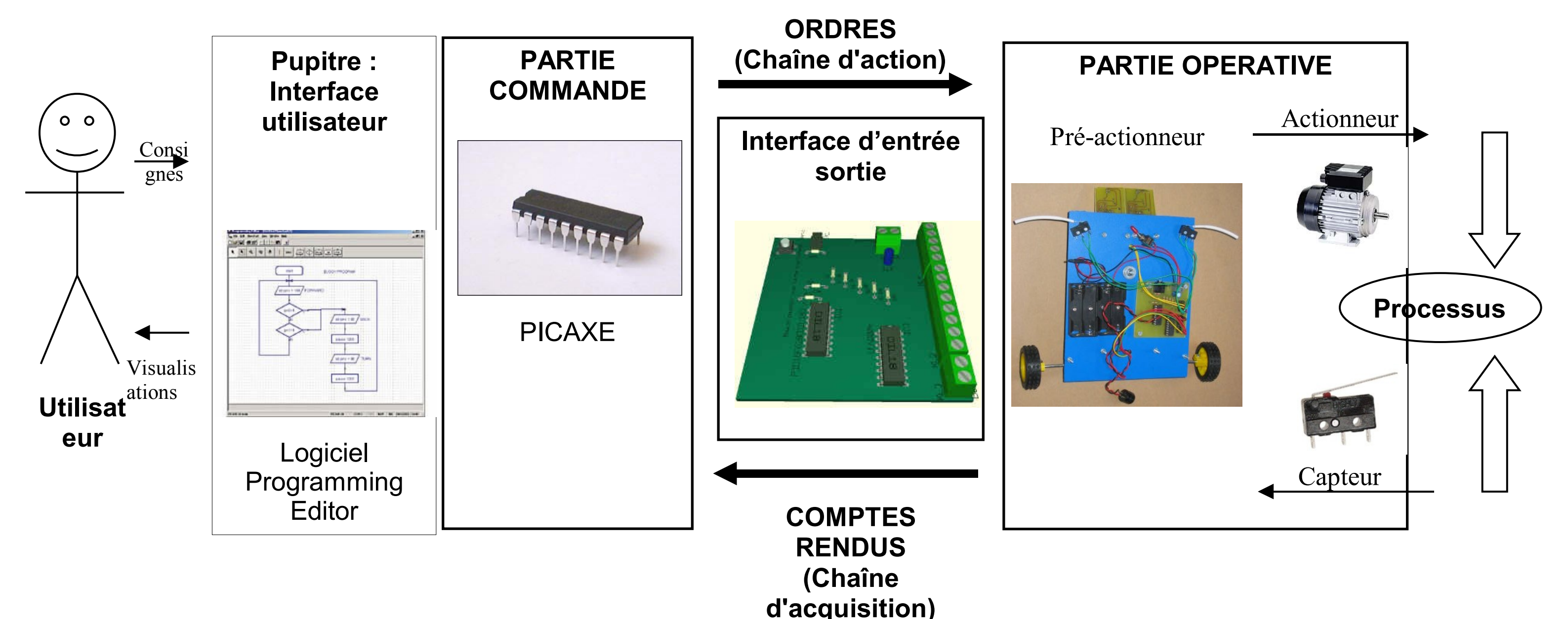


AUT : Programming Editor

MODE D'EMPLOI



DEFINITIONS :

•Automatisme : Un automatisme est une machine conçue pour effectuer d'elle même un ensemble d'opérations.

•Partie commande : Partie qui donne les ordres et recoit les comptes rendus.

•Partie opérative : Partie qui execute les ordres à l'aide des actionneurs et qui envoie les comptes rendus des informations fournies par les capteurs.

•Capteur : Organe qui reçoit une grandeur physique et la transforme en un signal mesurable.

•Actionneur : c'est un élément de la partie opérative qui convertit un ordre de la partie commande en une action physique du système automatisé.

La partie commande

Dans notre cas, la partie commande est un PICAXE, c'est à dire un composant electronique programmable en langage machine. Le langage machine est une suite de bits (0 et 1) qui impose un fonctionnement précis au composant programmable. Le PICAXE peut être programmé à volonté par le biais d'un logiciel facilitant l'élaboration du programme,

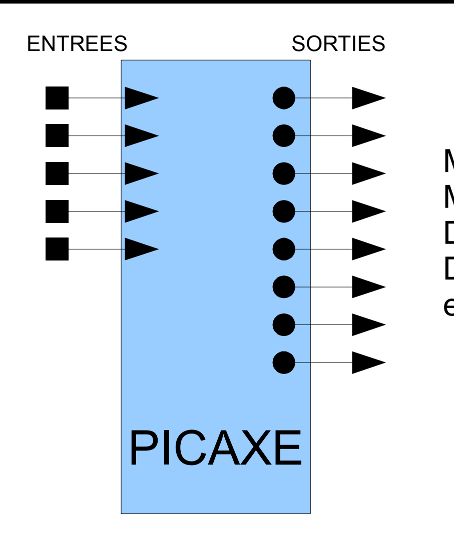
Le PICAXE est composé de broches. Ces broches permettent de communiquer sous forme binaire. Une broche peut être dans 2 états, à l'état haut (+5V) et à l'état bas (0V).

La partie commande va envoyer des ordres sous la forme de tensions (0V ou +5V) qui vont indiquer aux actionneurs de se mettre en marche ou de s'arrêter.

De la même façon, les capteurs vont communiquer avec la partie commande sous forme de tensions (0V et +5V). Par exemple un capteur detectant une grandeur physique mettra une broche de la partie commande à l'état haut (+5V).

CAPTEURS :

Capteur de ligne droit
Capteur de ligne centre
Capteur de ligne gauche
Antenne droite
Antenne gauche
etc.



ACTIONNEURS :

Moteur droit
Moteur gauche
DEL droite
DEL gauche
etc.

La partie opérative

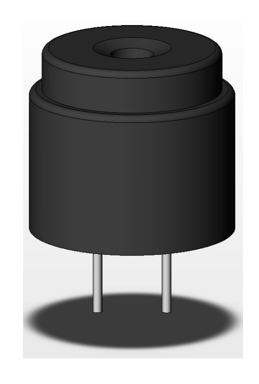
C'est la partie qui effectue les opérations. Les ordres venant de la partie commande sont traités par un pré-actionneur qui adapte le 0V et le 5V pour pouvoir mettre en marche un moteur.
Ainsi deux signaux venants de deux sorties de la partie commande sont connectés par l'intermédiaire du pré-actionneur sur les pôles du moteur. Ces signaux pourront alors mettre en marche le moteur, l'arrêter ou le mettre en marche arrière.

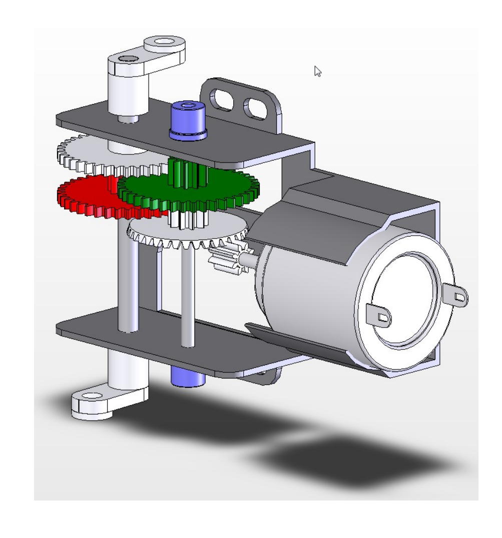
Les comptes rendus des capteurs sont transmis après adaptation en 0V ou +5V.

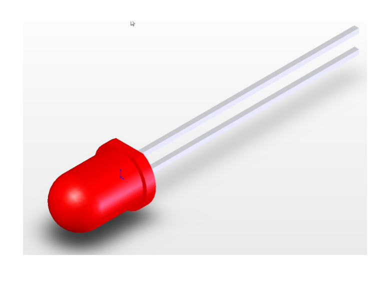
Les actionneurs

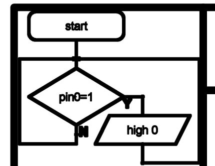
Un actionneur crée un phénomène physique à partir des ordres de la partie commande. Par exemple, un motoréducteur est mis en rotation s'il est branché par l'intermédiaire d'un pré-actionneur, à deux sorties de la partie commande.
Exemple d'actionneurs :

•Moteurs
•Motoréducteur (moteur et engrenages pour reduire la vitesse de rotation)
•DEL (Diode Electro Luminescente)
•Buzzer
•Relai electromagnétique







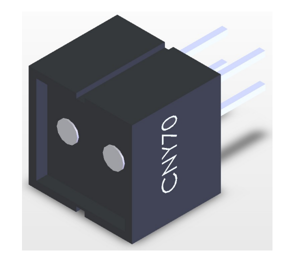


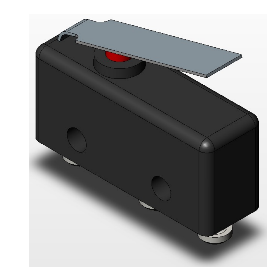
Les capteurs

Un capteur transforme un phénomène physique pour l'envoyer comme compte rendu à la partie commande. Par exemple, un interrupteur capte un contact et transmet un état haut (+5V) à la partie commande.
Exemple de capteurs :

- Interrupteurs
- Contact de fin de course
- Capteur de luminosité
- Capteur infrarouge
- ILS (Interrupteur à lame souple)

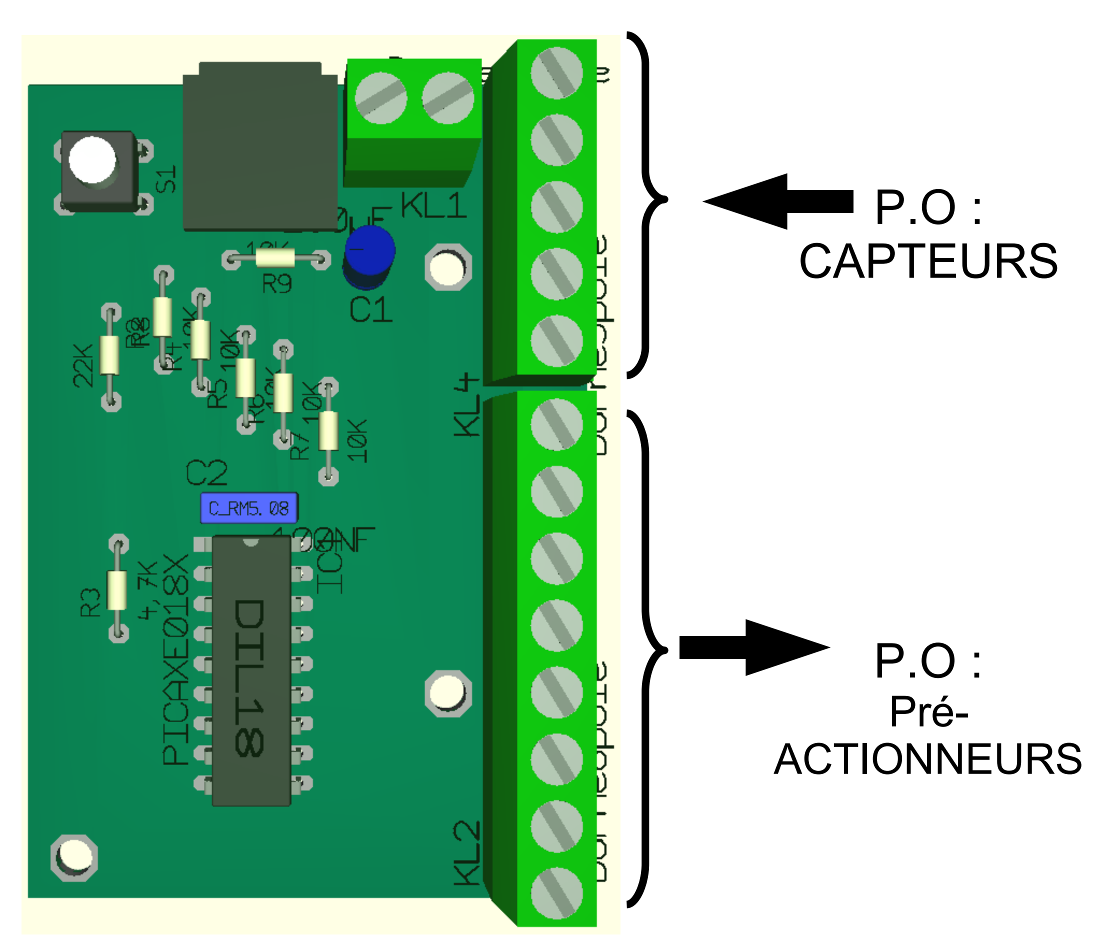
etc.

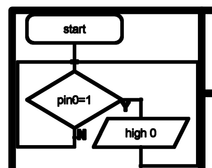




L'interface entrée sortie

Cette interface permet d'établir les connexions entre la partie commande et la partie opérative.
Dans notre cas il s'agit d'une carte électronique avec des borniers à vis pour connecter les fils allant aux pré-actionneurs et aux capteurs.



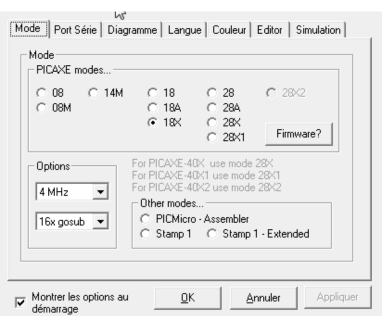


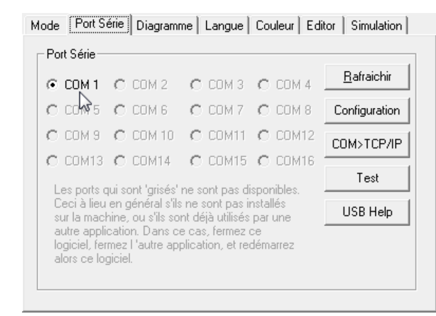
AUT : Programming Editor

Fonctionnement du logiciel

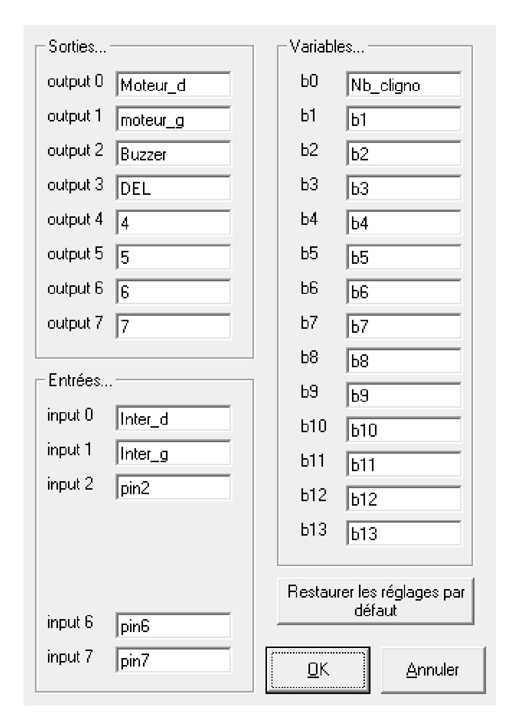
Choix du modèle de PICAXE et du port COM

•Démarrer le logiciel Picaxe Programming Editor (***Menu démarrer*** – ***Programmes*** – ***Revolution Education*** – ***Picaxe Programming Editor***)
•Les options s'affichent au démarrage. Dans l'onglet ***Mode***, choisir le modèle de PICAXE équipant votre maquette.
•Dans l'onglet ***Port série***, choisir le port qui servira à la programmation, le plus souvent COM1





Paramétrage : Changer le nom des entrées et des sorties (ex. Output 0 → MOTEUR_D)



Dans le menu ***Diagramme*** cliquer sur ***Table des symboles pour les diagrammes...*** On peut ensuite choisir des noms plus parlant pour désigner ce qui est branché sur les entrées ou les sorties de l'interface.

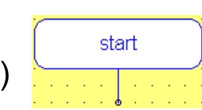
Par exemple Moteur_d pour le moteur droit est un nom plus pratique que output0. Les variables permettent d'enregistrer des informations, il faut également leur donner des noms plus parlant, cela fera gagner du temps lors de la conception de l'organigramme de programmation.

Il faut changer les noms des Sorties, des Entrées et des Variables que l'on va utiliser dans la programmation.

Élaborer un organigramme

Pour débuter un diagramme cliquer sur

Un programme commence par l'étape START (Début)



Un organigramme est composé d'étape de plusieurs sortes :

Les étapes ACTIONS — ex : Mettre un actionneur en marche

Les étapes TEST — ex : Tester l'état d'un capteur

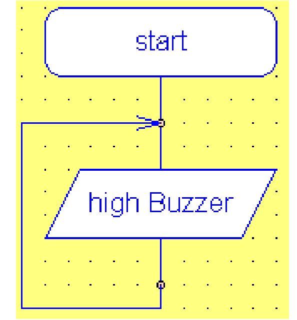
Les autres étapes — ex : Attendre pendant quelques secondes

Les connecteurs ronds permettent de connecter un lien (une ligne) vers l'étape suivante ou directement l'étape suivante.

Utiliser un actionneur

Pour utiliser un actionneur, il faut d'abord donner des noms parlant aux entrées sorties comme montré plus haut (on peut aussi utiliser le numéro de la sortie sur lequel il est connecté).

Si un buzzer est branché (par l'intermédiaire d'un pré-actionneur) sur la sortie 2, pour le faire sonner il suffit de dessiner l'organigramme suivant :



Cet organigramme peut se traduire de la façon suivante :

Au départ mettre la sortie Buzzer à l'état haut (high), puis revenir au départ et recommencer.

La ligne se dessine avec l'icône

Utiliser un capteur

Pour utiliser un capteur, il faut d'abord donner des noms parlant aux entrées sorties comme montré plus haut (on peut aussi utiliser le numéro de la sortie sur lequel il est connecté).

Si un interrupteur est branché sur l'entrée 0, pour le faire sonner un buzzer (sortie 2) si l'interrupteur est fermé il suffit de dessiner l'organigramme suivant :

Cet organigramme peut se traduire de la façon suivante :

Au départ tester si l'entrée Inter_d est égale à 1, c'est à dire si elle est à l'état haut (high).
•Si oui (Y pour yes) alors mettre la sortie Buzzer à 1 (c'est à dire faire sonner le buzzer) puis revenir au départ et recommencer.
•Si non (N pour no) alors mettre la sortie Buzzer à 0 (c'est à dire ne pas faire sonner le buzzer) puis revenir au départ et recommencer.

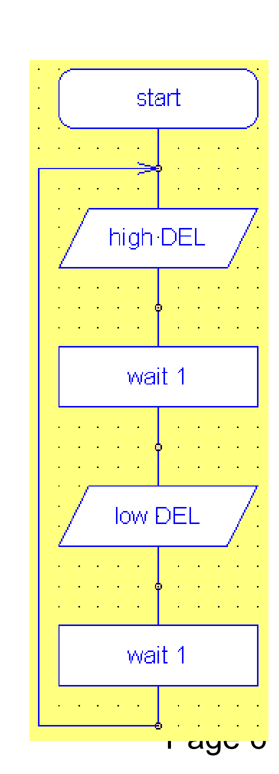
Utiliser une temporisation

Une temporisation permet de placer le programme en attente pendant un certain temps. Cela permet par exemple de faire clignoter une DEL à une période voulue.

Les temporisations PAUSE sont en mili-seconde. Pause 100 signifie attendre 100 ms.

Les temporisations WAIT sont en seconde. Pause 1 signifie attendre 1s.

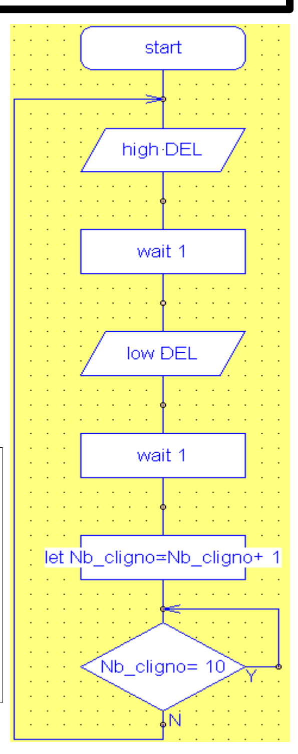
Si une DEL est branchée (par l'intermédiaire d'un pré-actionneur) sur la sortie 3, pour l'allumer pendant 1s puis l'éteindre pendant 1s et recommencer il suffit de dessiner l'organigramme suivant :



Cet organigramme peut se traduire de la façon suivante :

Au départ mettre la sortie 0 à l'état haut (high), c'est à dire allumer la DEL, puis faire une pause de 1000ms (1s), ensuite mettre la sortie 0 à l'état bas (low), c'est à dire éteindre la DEL, puis revenir au départ et recommencer.

Utiliser une variable



Une variable permet par exemple de compter un évènement. Cela permet par exemple de faire clignoter une DEL qu'un nombre limité de fois.

Si une DEL est branchée (par l'intermédiaire d'un pré-actionneur) sur la sortie 0, pour la faire clignoter 10 fois il suffit de dessiner l'organigramme suivant :

Cet organigramme peut se traduire de la façon suivante :

Au départ, comme dans l'exemple sur la temporisation, faire clignoter la DEL (elle est allumée 1s puis éteinte 1s), ensuite on ajoute 1 au contenu de la mémoire Nb_cligno (Nb_clign = Nb_cligno + 1 ajoute 1 à Nb_cligno et met le résultat dans Nb_cligno, on appelle cette technique incrémenter Nb_cligno), on fait de nouveau clignoter la DEL, on ajoute de nouveau 1 donc Nb_cligno = 2, on recommence, Nb_cligno = 3 etc. Quand le test Nb_cligno = 10 est vrai alors le programme ne peut plus revenir au départ à cause du test qui passe par Y.

Paramétrage : Définir les sorties actives pour les déplacements

Pour faciliter la programmation de déplacement le logiciel a prévu des étapes ACTION spéciales pour les déplacements. Pour les utiliser il faut configurer sur quelles sorties sont branchés les moteurs.

•Dans le menu ***Affichage*** cliquer sur ***Options***. Dans l'onglet ***Diagramme***, cliquer sur le bouton ***Déplacement*** et sélectionner les sorties actives pour :

- Avancer (Forward)
- Reculer (Back)
- Tourner à gauche (Left)
- Tourner à droite (Right)

Déplacement

	7	6	5	4	3	2	1	0	
Forward	☐	☐	☐	☐	☐	☑	☐	☑	5
Back	☐	☐	☐	☐	☑	☐	☑	☐	10
Left	☐	☐	☐	☐	☐	☐	☐	☑	1
Right	☐	☐	☐	☐	☐	☐	☑	☐	2

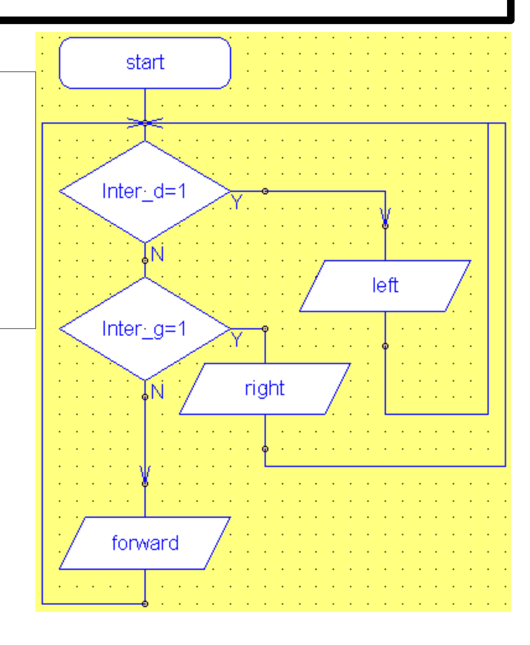
Exemple :
La borne + du moteur droit branchée sur la sortie 0
La borne - du moteur droit branchée sur la sortie 1
La borne + du moteur gauche branchée sur la sortie 2
La borne - du moteur gauche branchée sur la sortie 3

Alors pour avancer il faut que les sorties reliées aux bornes + soient à l'état haut (+5V) et que les bornes – soient à l'état bas (0V), c'est à dire cochées dans cette fenêtre.

Il faut donc cocher 0 et 2 pour la ligne Forward
Il faut cocher 1 et 3 pour Back
Il faut cocher 0 pour Left (seul le moteur droit tourne) ou 0 et 3 (le moteur droit avance le moteur gauche recule
Il faut cocher 2 pour Right (seul le moteur gauche tourne) ou 2 et 1 (le moteur droit avance le moteur gauche recule

Déplacement

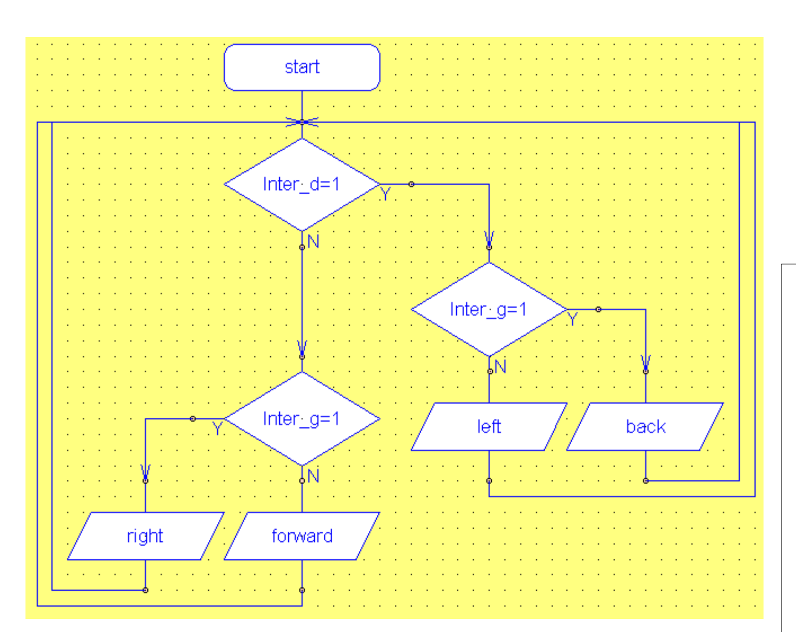
	7	6	5	4	3	2	1	0	
Forward	☐	☐	☐	☐	☐	☑	☐	☑	5
Back	☐	☐	☐	☐	☑	☐	☑	☐	10
Left	☐	☐	☐	☐	☑	☐	☐	☑	9
Right	☐	☐	☐	☐	☐	☑	☑	☐	6



Tourner à droite si obstacle à gauche, tourner à gauche si obstacle à droite sinon avancer

Au départ tester si l'entrée Inter_d = 1, c'est à dire si il y a un obstacle à droite.
Si oui (Y pour yes) alors tourner à gauche puis revenir au départ et recommencer.
Si non (N pour no) alors tester s'il y a un obstacle à gauche c'est à dire si Inter_g = 1,

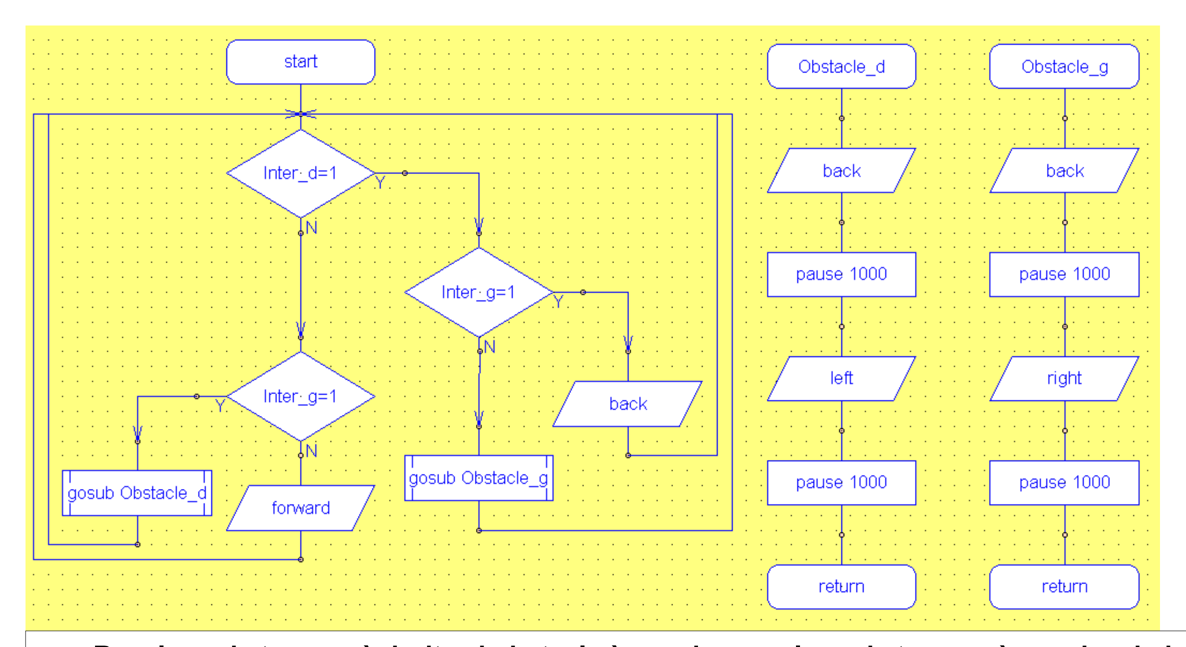
- Si oui tourner à gauche, puis revenir au départ et recommencer.
- Si non avancer



Tourner à droite si obstacle à gauche, tourner à gauche si obstacle à droite, reculer si obstacle à droite et à gauche, sinon avancer

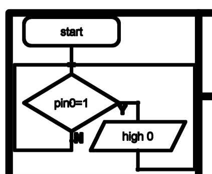
Au départ tester si l'entrée Inter_d = 1, c'est à dire si il y a un obstacle à droite.
•Si oui (Y pour yes) tester si il y a un obstacle à gauche, si non alors tourner à gauche puis revenir au départ et recommencer, si oui alors il y a un obstacle à droite et à gauche donc reculer.
•Si non (N pour no) alors tester s'il y a aun obstacle à gauche c'est à dire si Inter_g = 1,

- Si oui tourner à droite, puis revenir au départ et recommencer.
- Si non avancer



Reculer puis tourner à droite si obstacle à gauche, reculer puis tourner à gauche si obstacle à droite, reculer si obstacle à droite et à gauche, sinon avancer

Ce programme est le même que le précedent mais il utilise des sous programmes (Obstacle_g et Obstacle_d) qui permettent de simplifier l'organigramme principal. En cas d'obstacle, un sous programme est appelé (gosub), il fait d'abord reculer le robot puis le fait tourner pendant 1s avant de retourner au programme principal (return).



AUT : Programming Editor

Table des symboles

- Outil de sélection
- Sélection de plusieurs éléments
- Loupe
- Dessin de ligne reliant les étapes
- Revenir au menu précédent

- Accéder au étapes de mouvement
- Avancer
- Reculer
- Tourner à droite
- Tourner à gauche
- S'arrêter

- out : Accéder aux étapes ACTIONS
- high : Mettre une sortie à 1 (état haut)
- low : Mettre une sortie à 0 (état bas)
- toggle : Inverser une sortie

- delay : Accéder aux temporisation
- pause : Pause en ms
- wait : Pause en seconde

- if : Accéder aux étapes TEST
- pin : Teste une entrée sortie à gauche
- pin : Teste une entrée sortie à droite

- start : Étape de départ, elle est obligatoire
- low 0 : Étape ACTION mettre à 0 une sortie
- high 0 : Étape ACTION mettre à 1 une sortie
- toggle 0 : Étape ACTION inverser une sortie
- forward : Étape ACTION avancer
- back : Étape ACTION reculer
- left : Étape ACTION tourner à gauche
- right : Étape ACTION tourner à droite
- halt : Étape ACTION s'arrêter

- pause 1000 : Étape Temporisation
- pin0=1 (Y / N) : Étape TEST si pin0 = 1 ?